UNIVERSIDAD TECNOLOGICA
NACIONAL
FACULTAD REGIONAL
RESISTENCIA

INGENIERÍA EN SISTEMAS DE INFORMACIÓN

MATERIA: SISTEMAS DE GESTIÓN DE BASES DE DATOS

NIVEL: 3

CICLO LECTIVO: 2024 – 2DO. CUATRIMESTRE

DOCENTES:

PROFESOR: I.S.I. ANDRÉS PABLO FANTÍN

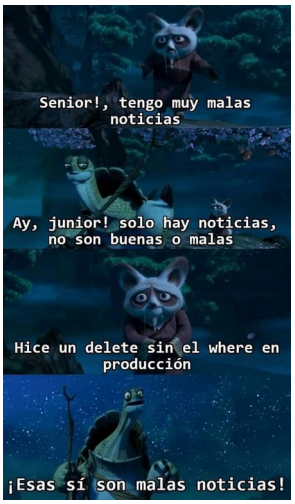
J.T.P.: I.S.I. JUAN CARLOS FERNÁNDEZ

AUXILIAR ADSCRIPTA: LUCIANA CAMPESTRINI

CARPETA DE TRABAJOS PRÁCTICOS

Trabajo Práctico N°3

Parte 1: Lenguaje DML



- Todo el proceso de escritura de las sentencias debe hacerse obligatoriamente **con línea de comandos** (en otras palabras, NO DEBEN USARSE LOS ASISTENTES GRÁFICOS).
- El producto de la guía serán scripts (archivos con extensión .sql) con las sentencias requeridas, especificando en cada caso con comentarios válidos en SQL a qué ejercicios corresponde la sentencia. Se acompañarán los scripts con una captura de pantalla de la terminal de comandos del SO o una pestaña de consultas de MySQL Workbench donde se constaten los resultados de la ejecución de las consultas.

Ejercicio 1: Sobre la base de datos de ejemplo *Sakila* provista por MySQL (https://dev.mysql.com/doc/sakila/en/), ejecute las instrucciones DML que realicen las siguientes acciones. Prevea que todas las tablas tengan datos cargados, para lo que cuenta con:

- Archivos descargables (modelo en MySQL Workbench, script de creación de esquemas y script de carga de datos): https://downloads.mysql.com/docs/sakila-db.tar.gz
- Instrucciones de instalación: https://dev.mysql.com/doc/sakila/en/sakila-installation.html

CONSULTAS DE INSERCIÓN, ACTUALIZACIÓN Y BORRADO

1.1. Inserte un nuevo **cliente**.
1.2. Agregue un nuevo **empleado** a la base.
1.3. Inserte los datos de un **cliente** existente (*nombre* y *apellido*) pero con un nuevo código (*customer_id*).
1.4. Borre todos los registros de la tabla **pelicula_actor** e inserte para todas las películas cargadas un registro asociándola con el actor con el menor id (*actor_id*).
1.5. Actualice el actor relacionado a una de las películas por el id de la actriz “JENNIFER DAVIS”.
1.6. Actualice la fecha de un alquiler que usted seleccione por 23/12/2004
1.7. Debe realizarse un descuento del 10% en los pagos que fueron hechos el mismo día del alquiler.

1.8. Actualice el código de empleado de los alquileres que tengan asignado al empleado 1 por el empleado 3.
1.9. Borre todos los alquileres anteriores al 30/05/2005.
1.10. Elimine a todos los empleados que no tengan cargado su *email*.
1.11. Insertar un nuevo empleado con el número inmediato consecutivo al máximo existente.
1.12. Insertar una nueva categoría con el número inmediato consecutivo al máximo existente sin utilizar la función MAX ni la propiedad “Auto increment” de la clave primaria.
1.13. Insertar los datos de la tabla **cliente** en una nueva tabla, borre los datos de la tabla **cliente**. En la nueva tabla realice una actualización de los códigos de cliente incrementándolos en uno. Posteriormente reinsértelos en la tabla **cliente** y vuelva a la normalidad los códigos.
1.14. Las compañías telefónicas han decidido (por falta de números telefónicos!!), que todas las líneas deben agregar un 9 como primer número. Realice la actualización correspondiente en los teléfonos (*phone*) registrados para cada dirección (**address**).
1.15. Debido a una promoción, se incrementa en dos los días de alquiler (*rental_duration*) de las películas con una tarifa (*rental_rate*) menor a 2.5.

CONSULTAS SIMPLES

1.16. Obtenga un listado de todos los empleados de nombre Jon.
1.17. Se desea obtener la cantidad de alquileres con fecha posterior a 30/06/2005.
1.18. Se necesita conocer la cantidad total de alquileres de cada película.
1.19. Liste todos los alquileres hechos entre el 20/12/2005 y el 10/01/2006.
1.20. Muestre los alquileres en los cuales la fecha de devolución (*return_date*) es posterior a 2 días desde la fecha de alquiler (*rental_date*).
1.21. Obtenga la cantidad de clientes diferentes que alquilaron películas después del 31/01/2006.
1.22. Obtener los datos del último alquiler existente
1.23. Obtener los apellidos de los empleados que se encuentren repetidos.
1.24. Obtener un listado con la cantidad de alquileres por fecha.
1.25. Obtener el promedio de días que las películas tardan en devolverse (diferencia entre fecha de devolución y fecha de alquiler). Cree una vista con esta consulta que se llame vw_promedio_dias.
1.26. Obtener los empleados que hayan cobrado más de 10 pagos. Cree una vista con esta consulta que se llame vw_empleados_pagos.

CONSULTAS MULTITABLA

1.27. Listar los clientes cuyas direcciones principales estén sobre calles que empiecen con A (tener en cuenta el formato de dirección cargado), indicando nombre, domicilio y teléfono. Hacer una versión en la que aparezcan sólo los que tienen teléfono, y hacer otra en la que aparezca sólo los clientes con domicilio en calles que empiecen con A y que no tienen ningún teléfono.
1.28. Listar los alquileres mostrando día, nombre del cliente, título de la película y el nombre del empleado que atendió.
1.29. Listar los alquileres, la cantidad días que estuvo prestada la película y el monto pagado.
1.30. Listar los alquileres y los nombres de clientes que pagaron más de 250 en total.
1.31. Listar los clientes que fueron atendidos alguna vez por el empleado “Jon Stephens”.
1.32. Listar los clientes que realizaron más de un alquiler el mismo día.
1.33. Listar los nombres de los empleados y la cantidad de pagos que atendieron para aquellos con un monto mayor a 5.
1.34. Crear una vista que liste la cantidad total de alquileres por categoría de películas.

CONSULTAS ANIDADAS – SUBCONSULTAS: Los ejercicios que se proponen a continuación se pueden resolver de varias maneras, intente resolverlos utilizando subconsultas ya que de eso trata

el tema. Además un mismo ejercicio lo puede resolver de diferentes maneras utilizando distintos tipos de condiciones y/o estrategias (CTE, consultas derivadas laterales, etc.), así un ejercicio se puede convertir en dos o tres ejercicios. Resuelva al menos 5 (cinco) consultas con al menos 2 (dos) alternativas diferentes.

1.35. Listar las películas en las que actuó alguna vez la actriz “JENNIFER DAVIS”.
1.36. Listar los códigos de clientes que hayan alquilado más de 50 películas distintas.
1.37. Listar los nombres de clientes que alquilaron películas en 2004 pero no lo hicieron en 2005.
1.38. Listar los empleados (código, nombre y apellido) que atendieron más de 10 alquileres y por los que el monto total cobrado fue más de 50.
1.39. Listar los actores que actuaron en al menos 2 películas distintas, las cuales fueron alquiladas más de 30 veces.
1.40. Listar el nombre, apellido y código de aquellos clientes que se han retrasado en la devolución más de un 40% de las veces que alquilaron.
1.41. Crear una vista que liste las películas que tienen disponibles (sin alquilar) más del 20% de sus existencias (en la tabla **inventory**).
1.42. Crear una vista que liste el nombre y el código de aquellos clientes que hayan alquilado el día en que se registró la mayor cantidad de alquileres.
1.43. Listar el código y nombre de los clientes que hayan hecho alquileres en la fecha más antigua registrada.
1.44. Listar los clientes que alquilaron entre los años 2005 y en el 2006.
1.45. Listar las películas que tienen menos de 5 ejemplare en existencia y que han generado ingresos totales por más de 200.
1.46. Listar los nombres y códigos de clientes que alquilaron todas las películas de acción (categoría “Action”).
1.47. Listar los empleados que atendieron alquileres de todas las categorías de películas.

Ejercicio 2: Utilizando el Ejercicio 1 del TP 2 realice las siguientes acciones:

2.1. Agregue un campo *sueldo* a la tabla **PERSONAL** e inserte masivamente datos en todas las tablas utilizando para ello algún generador de datos aleatorios (por ejemplo https://generatedata.com/es). Para ello deberá generar como mínimo 100 filas en más de una tabla e importar al menos una vez datos en los siguientes formatos:
- CSV con delimitador “;”
- SQL
2.2. Otorgue un aumento del 10% a los sueldos del personal con un sueldo inferior a los $150.000 y del 5% a aquellos con un sueldo entre $150.000 y $200.000 inclusive.

Ejercicio 3: Sean las siguientes tablas de una base de datos de una inmobiliaria:

Inquilino(CUIT,Nombre,Apellido)
Alquiler(CUIT,Id_casa,Deuda)
Nota: Deuda >=0 (si es 0, no hay deuda)
Telefonos(CUIT,Fono)
Dueño(CUIT,Nombre,Apellido)
Casa(Id_casa,CUIT,Nro,Calle,Ciudad)

Cree una base de datos con las tablas correspondientes y escriba consultas sql que contesten las siguientes preguntas, cargando previamente algunos registros de manera tal que devuelvan resultados:
3.1. Los arrendatarios que arriendan la casa ubicada en la calle Carrera nº 1024, Corrientes.
3.2. ¿Cuánto le deben a María Pérez?
3.3. ¿Cuál es la deuda total para cada dueño?
3.4. Liste todas las personas de la base de datos
3.5. Indique los dueños que poseen tres o más casas.
3.6. Liste los dueños que tengan deudores en todas sus casas.
3.7. Cree una vista (y luego compruebe su funcionamiento) que se llame “vw_estadisticas” que entregue estadísticas sobre los arrendatarios por casa. Liste:

- El promedio.
- La varianza.
- El máximo.
- El mínimo.

Parte 2: Programación en Bases de Datos

EJERCICIO 1: Escenario de Alquiler de Películas

PROGRAMAS ALMACENADOS (en MySQL utilizar la rutina más adecuada según el caso: procedimiento o función)

1. Crear un programa almacenado que tenga como parámetro de entrada un número entero (código de empleado: *staff_id*) y devuelva el monto total en pagos cobrados por ese empleado.
2. Crear un programa almacenado para ingresar nuevos actores, que tenga como parámetros de entrada los valores de *first_name* y *last_name*.
3. Crear un programa almacenado que liste los nombres de los actores con apellidos que comiencen con una cadena de texto determinada por un parámetro de entrada (Valor predeterminado: comienza con A).
4. Crear un programa almacenado que realice una inserción de 26 actores en forma secuencial de la forma:

Nom_Actor1	Ape_Actor1
Nom_Actor2	Ape_Actor2
Nom_Actor3	Ape_Actor3

5. Crear un programa almacenado que inserte valores en la tabla de países (*country*) pero que además de estos parámetros requiera nombre de usuario y contraseña. Estos dos parámetros deben ser validados en la tabla **staff** con los campos *username* y *password*.

TRIGGERS

6. Crear un trigger sobre la tabla **category** que en caso de que se borre una fila de ésta elimine las respectivas filas de la tabla **film_category**.

7. Crear un trigger que en caso de UPDATE del *actor_id* de la tabla **actor**, haga el correspondiente update en la tabla **film_actor**.
8. Crear un trigger que en caso de inserción sobre la tabla **city** verifique la existencia del código de país correspondiente y en caso de no encontrarlo no realice el proceso.
9. Crear un trigger en donde, si se insertan valores en la tabla **payment**, si el monto ingresado es mayor a 50, inserte en una tabla llamada AUDITORIA (previamente creada) los valores insertados.

EJERCICIO 2: Bares y Cervezas

Dadas las siguientes relaciones

CERVEZAS *(nombre, fabricante)*
PRECIOS *(Bar, Cerveza, Precio)*
Cerveza → ***CERVEZAS(****nombre****)***

1. Cree las tablas correspondientes
2. Inserte datos de prueba
3. Cree un disparador que asegure que cualquier cerveza añadida a la lista de precios figure ya en la lista de cervezas. Para ello, en caso de que la cerveza no aparezca, añadirá una nueva entrada a la lista dejando el fabricante a NULL.
4. Suponga la siguiente relación: ***SUPER_PRECIOS*** *(Bar, Cerveza, Precio, Fecha).* Defina un disparador para guardar en la tabla correspondiente (SUPER_PRECIOS) una lista de aquellos bares que suban el precio de alguna de sus cervezas en un precio superior a $150, así como la marca de la cerveza, el nuevo precio asignado y la fecha en la que se realizó la actualización.

EJERCICIO 3: Histórico Socios del Videoclub

Dada la siguiente relación:

SOCIO *(num_soc, nombre direccion, telefono)*

1. Cree la tabla correspondiente.
2. Inserte algunos valores de prueba en la tabla recién creada.
3. Se desea mantener la información de los socios aunque estos se den de baja, para lo que se crea una tabla SOCIO_BAJA, que contiene los datos de socio y la fecha de baja, que se actualizará cada vez que se borre un socio, por tanto:
   3.1. Cree la tabla correspondiente a la relación SOCIO_BAJA
   SOCIO_BAJA *(num_soc, nombre direccion, teléfono, fecha_baja)*
   3.2. Defina un disparador que se encargue de realizar la funcionalidad especificada. Documente los resultados con una copia de pantalla.
   3.3. Elimine algunos datos de la tabla SOCIO con el fin de comprobar el correcto funcionamiento del disparador creado.

EJERCICIO 4: Gestión de Almacenes

Dadas las siguientes relaciones:

PRODUCTO *(cod_prod, descripción, proveedor, unid_vendidas)*
ALMACEN *(cod_prod_s, stock, stock_min, stock_max)*

1. Cree las tablas correspondientes.
2. Inserte algunos valores de prueba en la tabla ALMACEN

3. Crear los disparadores necesarios para proporcionar la siguiente funcionalidad:
   3.1. Se desea mantener actualizado el stock del ALMACEN cada vez que se vendan unidades de un determinado producto.
   3.2. Cuando el stock esté por debajo del mínimo lanzar un mensaje de petición de compra. Se indicará el número de unidades a comprar, según el stock actual y el stock máximo
   3.3. Si el stock es menor que el stock mínimo permitido, se debe impedir la venta.
4. Inserte algunos datos de prueba en la tabla PRODUCTO que permitan comprobar el correcto funcionamiento de los disparadores anteriores. Documente los resultados con una copia de pantalla.

EJERCICIO 5: Base Multidimensional

Para la implementación de una base de datos que soporte análisis multidimensional se desea crear una tabla que almacene las diferentes características referidas a una fecha; esto es: día, día de la semana, mes, año, cuatrimestre… El esquema relacional de dicha tabla sería el siguiente:

TIEMPO *(fecha, dia, dia_Semana, mes, cuatrimestre, anio)*

1. Crear la tabla TIEMPO en la base de datos atendiendo al esquema relacional anterior; considerando el atributo fecha como clave primaria. El resto de los atributos no admiten valores nulos.
2. Crear un procedimiento almacenado que dado un entero obtenga en formato de texto el correspondiente mes. Por ejemplo:

call convertir_mes(1)
"Enero"

3. Crear un procedimiento almacenado que recibiendo un rango de fechas introduzca en la tabla de TIEMPO una tupla por cada día comprendido entre ambas fechas. Por ejemplo:

call insertarTiempos(*'01/01/2012','10/01/2012'*);

Después de la llamada al procedimiento "insertarTiempos" del ejemplo anterior, la tabla TIEMPOS debería contener la siguiente información:

Fecha	dia	Dia_Semana	mes	cuarto	año
01/01/2012	1	Jueves	Enero	Q1	2012
02/01/2012	2	Viernes	Enero	Q1	2012
03/01/2012	3	Sábado	Enero	Q1	2012
04/01/2012	4	Domingo	Enero	Q1	2012
05/01/2012	5	Lunes	Enero	Q1	2012
06/01/2012	6	Martes	Enero	Q1	2012
07/01/2012	7	Miércoles	Enero	Q1	2012
08/01/2012	8	Jueves	Enero	Q1	2012
09/01/2012	9	Viernes	Enero	Q1	2012
10/01/2012	10	Sábado	Enero	Q1	2012

4. Ejecutar el procedimiento creado para que genere las tuplas correspondientes a los años 2011 y 2012.

La tabla TIEMPO creada anteriormente la utilizaremos para fechar las ventas realizadas por una empresa dedicada a la venta de materiales informáticos. Por tanto el esquema relacional de la base de datos quedaría de la siguiente forma.

TIEMPO *(fecha, dia, dia_Semana, mes, cuatrimestre, anio)*
VENTA *(id_Venta,fecha, producto, unidades_vendidas, precio_unitario)*

5. Crear la tabla VENTA atendiendo al esquema relacional anterior; considerando el atributo id_venta como clave primaria. El resto de los atributos de la relación no pueden tomar valor nulo.
6. Crear un procedimiento/función almacenado llamado “registrar_venta” que recibiendo una fecha, un producto, el precio unitario del mismo y las unidades vendidas inserte los valores correspondientes en la tabla de ventas.
7. Utilizar el procedimiento almacenado creado anteriormente para realizar las siguientes ventas:

Fecha	producto	precio_unitario	unidades_vendidas
10/01/2011	Disco Duro 160 GB UDMA	100.00	4
17/01/2011	Disco Duro 250 GB UDMA	125.00	2
05/05/2011	Disco Duro 500 GB UDMA	150.00	10
15/05/2011	Disco Duro 750 GB UDMA	175.00	1
25/07/2011	Fuente Alimentación 1000W	55.20	4
02/08/2011	Memoria 1GB DDR 400	10.98	40
10/09/2011	Memoria 1GB DDR2 1066	22.65	2
04/12/2011	Memoria 2GB DDR3 800	32.15	3
05/12/2011	Memoria 1 GB Compact Flash	11.54	10
10/02/2012	Memoria 2 GB Compact Flash	19.87	1
11/05/2012	Memoria 8 GB Compact Flash	48.32	1
12/06/2012	Memoria 1 GB Secure Digital	14.52	15
14/08/2012	Memoria 2 GB Secure Digital	28.69	25
21/08/2012	Antena Wifi Cisco Omnidireccional	67.12	2
05/10/2012	Cable para antena 10 metros SMA	2.65	1
24/10/2012	Adaptador de Bluetooth 200 metros USB 2.0	14.32	1
26/11/2012	Adaptador de HomePlug Ethernet AV	23.16	10
05/12/2012	Adaptador de Infrarojos USB	47.88	8
19/12/2012	Hub 4 puertos USB 2.0	23.10	6

8. Crear un procedimiento “Mostrar_Estadísticas” que genere la siguiente información, considerando que la mayor venta se refiere a las ventas con mayor valor económico, no con mayor número de unidades vendidas:
    1. Valor total de las ventas.
    2. Valor total de las ventas en el año 2011.
    3. Valor total de las ventas en 2012.
    4. Listado ordenado de las ventas por día de la semana.
    5. Listado ordenado de las ventas por mes del año.
    6. Listado ordenado de la ventas por cuatrimestre del año.

9. Crear un evento programado para que diariamente, a las 0:00 horas, agregue un registro correspondiente a la fecha actual en la tabla **TIEMPO**.

Referencias

- Capítulo 13.2 “Data Manipulation Statements” del Manual de Referencia de MySQL: https://dev.mysql.com/doc/refman/8.4/en/sql-data-manipulation-statements.html.
- Documentación base de datos de ejemplo Sakila: https://dev.mysql.com/doc/sakila/en/
- Archivos de base de datos Sakila: http://downloads.mysql.com/docs/sakila-db.zip
- Capítulo 25 “Stored Programs and Views” del Manual de Referencia de MySQL: https://dev.mysql.com/doc/refman/8.4/en/stored-objects.html.
- Capítulo 13.6.4 “Variables en programas almacenados” del Manual de Referencia de MySQL: https://dev.mysql.com/doc/refman/8.4/en/stored-program-variables.html.
- Capítulo 9.4 “Variables definidas por el usuario” del Manual de Referencia de MySQL: https://dev.mysql.com/doc/refman/8.4/en/user-variables.html.
- Documentación de MySql Workbench: https://dev.mysql.com/doc/workbench/en/
- Foro de la Comunidad MySQL: http://forums.mysql.com
- Stack Overflow Business Solutions: https://stackoverflow.com/
- Database Management Systems (ISBN-10: 0072465638 | ISBN-13: 978-0072465631) de Raghu Ramakrishnan, U. de Wisconsin, EE.UU, tercera edición. Capítulo 5.
- Presentaciones de teoría.